Bifurcation Spiking Neural Networks (JMLR'21)

Shao-Qun Zhang, Zhao-Yu Zhang, and Zhi-Hua Zhou

LAMDA Group, Nanjing University, Nanjing, China

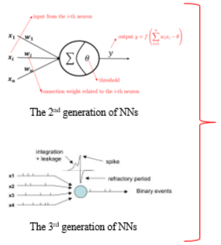
Contact

zhangsq@lamda.nju.edu.cn
zhangzhaoyu@lamda.nju.edu.cn
zhouzh@lamda.nju.edu.cn

Spiking Neural Networks

Neural Network Learning = Neural Network Model + Learning Algorithm

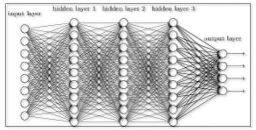
e.g., BP algorithm



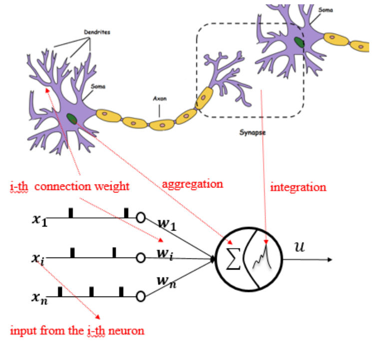
Neuron Model + Network Architecture

e.g., MP model — e.g., feed-forward

Spiking Neural Model



SNNs take into account the time of spike firing rather than simply relying on the accumulated signal strength in conventional neural networks, and thus offering the possibility for modeling time-dependent data. The fundamental spiking neural model is usually formulated as a first-order parabolic equation with many biologically realistic (i.e., internal) hyper-parameters. Thus, the performance of SNNs depends not only on determining the neural network architecture and training connection weights as well as conventional deep neural networks but also on the careful tuning of these internal hyper-parameters.



perfect Integrate-and-Fire equation (IF)
[Louis Lapicque, 1907]

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = \sum_{i=1}^{m} w_i x_i$$

Leaky Integrate-and-Fire equation (LIF)

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u + \sum_{i=1}^{m} w_i x_i$$

The "leak" term reflects the diffusion of ions through the membrane.

Other extended Integrate-and-Fire equation

Investigation from Dynamical Systems

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u + \sum_{i=1}^{m} w_i x_i$$

multiply u on both sides

$$\frac{\tau_m}{2}\frac{\mathrm{d}u^2}{\mathrm{d}t} = -u^2 + \left(\sum_{i=1}^{m} w_i x_i\right) u$$

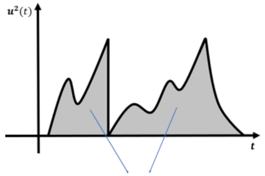
$$-\frac{2}{\tau_m}u^2 = \frac{\mathrm{d}u^2}{\mathrm{d}t} - \frac{2}{\tau_m}\left(\sum_{i=1}^{m} w_i x_i\right) u$$

energy function — energy estimation theorem

$$\mathcal{H}(u,t) \overset{\text{def}}{=} u^2 - \frac{2}{\tau_m}\int_t \left(\sum_{i=1}^{m} w_i x_i(t)\right) u(t)\,\mathrm{d}t$$

$$\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = 2u\left[\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{1}{\tau_m}\left(\sum_{i=1}^{m} w_i x_i\right)\right] = -\frac{2}{\tau_m}u^2$$

relies on τ_m, not on connection weight w_i



$\mathcal{H}(u,t)$ denotes the integral of motion u^2.

- $\frac{1}{\tau_m} < 0$, $\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} > 0$ energy increases
- $\frac{1}{\tau_m} = 0$, $\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = 0$ conservative system
- $\frac{1}{\tau_m} > 0$, $\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} < 0$ energy decreases

Theorem 1 Given the initial condition $u_0 = 0$, the dynamical system led by one layer of LIF neurons is a bifurcation dynamical system, and τ_m is the corresponding bifurcation hyper-parameter.

- The setting of τ_m should be adaptive to environment or data.

  LIF $\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u + \sum_{i=1}^{m} w_i x_i \quad \left(\frac{1}{\tau_m} > 0\right)$
  - good at handling the dissipative system
  - not suitable for handling the conservative and energy-diffusion systems
- The role of τ_m cannot be replaced by the connection weights.

  $\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = -\frac{2}{\tau_m}u^2$ relies on τ_m, not on connection weight w_i.
- Gradient-based approach ✗
  - Alternating gradient optimization

    lack of convergence guarantee
  - Pack τ_m and w_i as one parameter

    easy to fall into the problems of gradient explosion and vanishing

  LIF solution: (entangled)

  $$u(t) = \sum_{i=1}^{m} w_i \left[\int_{t'}^{t} \exp\left(\frac{t'-s}{\tau_m}\right) x_i(s)\,\mathrm{d}s\right]$$

  Gradients:

  $$\begin{cases} \nabla \tau_m \propto \sum_{i=1}^{m} w_i \left[\int_{t'}^{t} \exp\left(\frac{t'-s}{\tau_m}\right)(s-t')x_i(s)\,\mathrm{d}s\right], \\ \nabla w_i \propto \left[\int_{t'}^{t} \exp\left(\frac{t'-s}{\tau_m}\right) x_i(s)\,\mathrm{d}s\right] \end{cases}$$

  integration on the time interval
- Zero-order approach

  succeeds on an apposite initialization and larger computation and storage

Bifurcation Spiking Neural Networks

BSNN employs the self-connection architecture.

$$\frac{\partial u_k}{\partial t} = -\frac{1}{\tau_m}u_k + \sum_{j \neq k} \lambda_{kj} u_j + \frac{1}{\tau_m}\sum_{i=1}^{m} w_{ki} x_i$$

mutual promotion between neurons

energy derivative

$$\frac{\mathrm{d}\mathcal{H}_B}{\mathrm{d}t} = 2\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}^{\top} \begin{pmatrix} -\frac{1}{\tau_m} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & -\frac{1}{\tau_m} & \cdots & \lambda_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \cdots & -\frac{1}{\tau_m} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} >,\ =,\ \text{or}\ < 0 \ ?$$

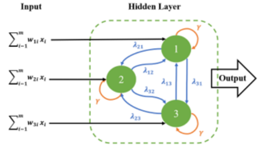


Illustration for one layer of BSNN

hyper-parameter τ_m + learnable matrix $\boldsymbol{\lambda}$

allow gradient calculation

We can convert the issue of searching for apposite hyper-parameters τ_m into a new problem of how to train the bifurcation parameters $\boldsymbol{\lambda}$.

Theorem 2 If the bifurcation parameters λ_{ij} are all great than 0, there are at most 2^{n-1} bifurcation solutions, where n is the number of hidden spiking neurons.

hyper-parameter τ_m + learnable matrix $\boldsymbol{\lambda}$ — 2^{n-1} solutions ⇒ $\frac{\mathrm{d}\mathcal{H}_B}{\mathrm{d}t}$ $>,\ =,\ \text{or}\ < 0$

- There are enough solutions for ensuring that BSNN is adaptive to environments or data.
- Numerical experiments show that the performance of BSNN is robustness to the setting of τ_m.

Experimental Results

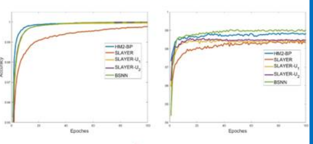
best accuracy

Table 2: The comparative performance of the contenders and BSNN.

Data Sets	Contenders	Accuracy (%)	Setting	Control Rate (γ)	Epochs
MNIST	Deep SNN (O'Connor and Welling, 2016)	97.80	28×28-300-300-10 ♠	-	50
	Deep SNN-BP (Lee et al., 2016)	98.71	28×28-800-10	-	200
	SNN-EP ♡	97.63	28×28-500-10	-	25
	HM2-BP (Jin et al., 2018)	98.84 ± 0.02	28×28-800-10	-	100
	SNN-L (Rezaabad and Vishwanath, 2020)	98.23 ± 0.07	28×28-1000-R28-10	-	-
	SLAYER (Shrestha and Orchard, 2018)	98.39 ± 0.04	28×28-500-500-10	-	50
	SLAYER-U_1 ♣	98.53 ± 0.03	28×28-500-500-10	-	-
	SLAYER-U_2	98.59 ± 0.01	28×28-500-500-10	-	-
	BSNN (this work)	**99.02 ± 0.04**	28×28-500-500-10	-0.21	50
N-MNIST	SKIM (Cohen et al., 2016)	92.87	2*28×28-10000-10	-	-
	Deep SNN-BP	98.78	2*28×28-800-10	-	200
	HM2-BP	98.84 ± 0.02	2*28×28-800-10	-	60
	SLAYER	98.89 ± 0.06	2*28×28-500-500-10	-	50
	SLAYER-U_1	99.01 ± 0.01	2*28×28-500-500-10	-	-
	SLAYER-U_2	99.07 ± 0.02	2*28×28-500-500-10	-	-
	BSNN (this work)	**99.24 ± 0.12**	2*28×28-500-500-10	-0.49	50
Fashion-MNIST	HM2-BP	88.99	28×28-400-400-10	-	15
	SLAYER	88.61 ± 0.17	28×28-500-500-10	-	50
	SLAYER-U_1	90.53 ± 0.04	28×28-500-500-10	-	-
	SLAYER-U_2	90.61 ± 0.02	28×28-500-500-10	-	-
	ST-RSBP (Zhang and Li, 2019)	90.00 ± 0.13	28×28-400-R400-10 ◇	-	30
	BSNN (this work)	**91.22 ± 0.06**	28×28-500-500-10	-0.32	50
EMNIST	eRBP (Neftci et al., 2017)	78.17	28×28-200-200-47	-	30
	HM2-BP	84.43 ± 0.10	28×28-400-400-10	-	20
	SNN-L	83.75 ± 0.15	28×28-1000-R28-10	-	-
	SLAYER	85.73 ± 0.16	28×28-500-500-47	-	50
	SLAYER-U_2	86.62 ± 0.03	28×28-500-500-47	-	50
	BSNN (this work)	**87.51 ± 0.23**	28×28-500-500-47	-0.37	50

♠: -300-300- denotes two hidden layers with 300 spiking neurons, while -800- is one hidden layer with 800 spiking neurons.
♡: SNN-EP (O'Connor et al., 2019) proposes an implementation for training SNN with equilibrium propagation.
♣: -U_1 and -U_2 indicate the alternating optimization algorithms with parametric control rates sampled from U_1 and U_2, respectively.
◇: R400 represents a recurrent layer of 400 spiking neurons.

faster convergence



robustness